

# Improved Decoding and Error Floor Analysis of Staircase Codes

Lukas Holzbaur, Hannes Bartz, and Antonia Wachter-Zeh\*

Institute for Communications Engineering, Technical University of Munich  
{lukas.holzbaur, hannes.bartz, antonia.wachter-zeh}@tum.de

**Abstract.** A low-complexity method for resolving stall patterns when decoding staircase codes is proposed. Stall patterns are the dominating contributor to the error floor in the original decoding method. Our improvement is based on locating stall patterns by intersecting non-zero syndromes and flipping the corresponding bits. The approach effectively lowers the error floor of decoding staircase codes. This allows for a new range of block sizes to be considered for optical communication at a certain rate or, alternatively, a significantly decreased error floor for the same block size. Further, an improved analysis of the error floor behavior is introduced which provides a more accurate estimation of the contributions to the error floor.

## 1 Introduction

Staircase codes were introduced by Smith *et al.* in [1] and are a powerful code construction for error-correction in optical communication systems. Staircase codes are based on a spatially-coupled binary BCH component code. With performance close to capacity for the binary symmetric channel (BSC) and decoder complexity lower than a comparable LDPC code, staircase codes provide a cost-efficient alternative to soft decision decoding of LDPC codes. As shown in [2], staircase codes perform well for a multitude of different parameters. However, the usability of staircase codes is limited by the requirement of optical communication systems to guarantee an error floor below  $10^{-15}$ , which allows small block sizes of the staircase code only at relatively low code rates.

Similar to trapping sets in decoding LDPC codes, certain constellations of errors, called *stall patterns*, cannot be resolved by the component codes of the staircase code. A strategy that enables the decoder to resolve stall patterns will improve the performance in the error floor region and potentially allow for more efficient decoding, as smaller block sizes can be used. For the structurally closely related product and half-product codes, several approaches for resolving stall patterns have been proposed. In [6,9] resolving of stall patterns by performing erasure decoding is considered. However, while requiring additional hardware in implementation, this approach offers no advantages in terms of stall pattern

---

\* A. Wachter-Zeh's work was supported by the Technical University of Munich—Institute for Advanced Study, funded by the German Excellence Initiative and European Union Seventh Framework Programme under Grant Agreement No. 291763. The authors would like to thank Gerhard Kramer and Frank Kschischang for the valuable discussions.

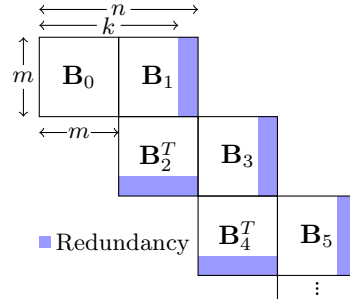
resolving capability and is even outperformed when comparing to approaches of lower complexity based on bit-flipping [7,8,4] for larger stall patterns. To evaluate the performance of a code with given parameters, an analysis of the error floor is required. While [7,4] offer an analysis based on exhaustive search, this work introduces a new analytical approach devised by relating the problem of counting stall patterns to the numerical problem of finding the number of binary matrices with certain row and column weight [5].

In this paper, we present an improved decoder for staircase codes, which is able to locate stall patterns and resolves many of them, by adapting the concept of bit-flipping as known for product codes. In particular, our method guarantees to correct *all* stall patterns when the number of involved columns and rows is each smaller than the minimum distance of the component BCH code and no undetected error events occur. Another contribution of this work is a new estimation of the error floor that is significantly more accurate than the one from [1]. Finally, we present simulation results for a staircase code with a quarter of the block size compared to the scheme of [1], which reaches a  $\text{BER}_{\text{out}} = 10^{-15}$  at  $\sim 1$  dB from BSC capacity at the cost of a tiny rate loss ( $\frac{236}{255}$  compared to  $\frac{239}{255}$ ). This scheme achieves a net coding gain (NCG) of 9.16dB at a  $\text{BER}_{\text{out}}$  of  $10^{-15}$ .

## 2 Staircase Codes

### 2.1 Encoding

Staircase codes are encoded block-wise, where each block  $\mathbf{B}$  is a binary  $m \times m$  matrix. The encoding procedure relies on a  $[n, k]$  BCH component code of error-correcting capability  $t$  and  $n = 2m$ . As in [1], we only consider extended BCH codes with a distance of  $d \geq 2t+2$  as component codes. The first block is initialized to all-zeros, every other block consists of  $m(k-m) = m \cdot k'$  information bits and  $m(n-k) = m(m-k')$  redundancy bits, giving the code rate  $R = \frac{k'}{m}$ . The encoding is done such that all rows of  $[\mathbf{B}_{i-1}^T \ \mathbf{B}_i]$  and all columns of  $\begin{bmatrix} \mathbf{B}_i \\ \mathbf{B}_{i+1}^T \end{bmatrix}$  are codewords of the BCH code, for all  $i \geq 1$ . An illustration is given in Fig. 1.



**Fig. 1.** Illustration of the structure of a staircase code.

### 2.2 Sliding-Window Decoding

Decoding is based on multiple iterations of hard-decision BCH decoders operating on a sliding window of appropriate size  $W$  in multiple iterations. A window comprised of  $\mathbf{B}_i$  to  $\mathbf{B}_{i+W-1}$  is decoded by first decoding the BCH codewords spanning  $[\mathbf{B}_i^T \ \mathbf{B}_{i+1}]$ , followed by the codewords spanning  $[\mathbf{B}_{i+1}^T \ \mathbf{B}_{i+2}]$ , until the last block  $\mathbf{B}_{i+W-1}$  of the window is reached. Then, the decoder returns to  $[\mathbf{B}_i^T \ \mathbf{B}_{i+1}]$  and the process is repeated. When no more errors are detected

in the last block of the window or a fixed maximum number of iterations is reached, the decoder declares  $\mathbf{B}_i$  as decoded, slides the window by one block and repeats the process for the new window comprised of  $\mathbf{B}_{i+1}$  to  $\mathbf{B}_{i+W}$ .

### 2.3 Stall Patterns and Known Error Floor Analysis

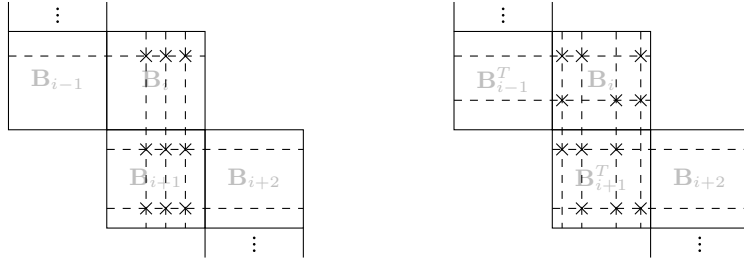
The  $[n, k]$  BCH component code has (unique) error-correcting capability  $t$ , i.e. if  $t + 1$  errors affect the codeword, the decoder is not able to resolve them.

A stall pattern for the staircase code is a set of erroneous bit positions such that each erroneous row and each erroneous column contains at least  $t + 1$  erroneous bits. The minimal number of rows  $K$  and columns  $L$  involved in such a pattern is  $K = L = t + 1$  and the minimal number of errors is  $\epsilon = (t + 1)^2$ . For such *minimal stall patterns* (compare Fig. 2), every intersection of an involved row and an involved column is an erroneous bit.

If more than  $t + 1$  rows or columns are part of the stall pattern, it is possible that not every bit in the intersection of involved rows and columns is in error. The weight of the error vectors of each involved row or column has to be at least  $t + 1$  and therefore, the number of errors  $\epsilon$  in a  $(K, L)$  stall pattern is bounded by:

$$\epsilon_{\min} \triangleq \max \{K, L\} \cdot (t + 1) \leq \epsilon \leq K \cdot L. \quad (1)$$

Fig. 3 shows a non-minimal  $(4, 4)$ -stall pattern with  $t = 2$  and  $\epsilon = \epsilon_{\min} = 12$ .



**Fig. 2.** Minimal stall pattern ( $t = 2$ )    **Fig. 3.** Non-minimal stall pattern ( $t = 2$ )

The error floor estimation given in [1] is based on the assumption that the dominating contributors to the error floor are stall patterns. It is estimated by enumerating the number of possible stall patterns and weighting each pattern with the probability that the corresponding positions are in error. A stall pattern is associated to the block  $\mathbf{B}_i$  with lowest index that contains at least one of its errors. The number of combinations of  $K$  rows and  $L$  columns such that the stall pattern belongs to a certain block is

$$A_{K,L} = \binom{m}{L} \cdot \sum_{a=1}^K \binom{m}{a} \cdot \binom{m}{K-a}. \quad (2)$$

Given the rows and columns, the number of different ways to distribute errors within their intersections is denoted by  $N_{K,L}^\epsilon$  is overestimated by (see [1]):

$$N_{K,L}^\epsilon \leq \binom{\min \{K, L\}}{t+1}^{\max \{K, L\}} \cdot \binom{K \cdot L - \epsilon_{\min}}{\epsilon - \epsilon_{\min}} =: \hat{N}_{K,L}^\epsilon, \quad (3)$$

With (2) and (3), the contribution of  $(K, L)$ -stall patterns to the  $\text{BER}_{\text{out}}$  in the error floor region can be overbounded by weighing each pattern with the probability that the corresponding positions are in error, to obtain

$$\sum_{\epsilon=(t+1) \cdot \max\{K,L\}}^{K \cdot L} \frac{\epsilon}{m^2} \cdot A_{K,L} \cdot \hat{N}_{K,L}^\epsilon \cdot (p + \xi)^\epsilon, \quad (4)$$

where  $p$  is the crossover probability of the BSC and  $\xi$  is an additional correction factor, see [1] for details.

### 3 Resolving Stall Patterns

#### 3.1 The Bit-Flip Operation

Assume that all errors that are not part of a stall pattern are resolved by the regular sliding-window decoding procedure (see Section 2.2) and hence only stall patterns remain.

In a minimal stall pattern, each involved row and column contains exactly  $t+1$  erroneous bits and therefore, results in a non-zero syndrome for the component code with distance  $d \geq 2t+2$ . Thus, a minimal stall pattern can be resolved by flipping each bit at the intersection of the words with non-zero syndromes.

**Definition 1 (Bit Flipping)** Consider a staircase code with the BCH component code  $\mathcal{C}$  and let  $\mathcal{A}_i = \{\mathbf{r}_0^i, \mathbf{r}_1^i, \dots, \mathbf{r}_{m-1}^i\}$  be the set of all received words corresponding to component codewords with redundancy bits in  $\mathbf{B}_{i+1}$ . Let  $\mathbf{S}_{\mathbf{r}_j^i}$  be the syndrome of  $\mathbf{r}_j^i$ . Define the elements of the vector  $\mathbf{b}^i \in \mathbb{F}_2^{m \times 1}$  for  $0 \leq j \leq m-1$  by:

$$\mathbf{b}^i(j) = \begin{cases} 1, & \text{if } \mathbf{S}_{\mathbf{r}_j^i} \neq \mathbf{0} \\ 0, & \text{else.} \end{cases} \quad (5)$$

Let  $\mathbf{M}_i = \mathbf{b}^{i-1} \times (\mathbf{b}^i)^T \in \mathbb{F}_2^{m \times m}$  be the **masking matrix** and let  $\mathbf{B}_i^{(z)}$  be block  $\mathbf{B}_i$  after  $z$  decoding iterations. Define the operation **bit-flip** as

$$\mathbf{B}_i^{(z+1)} = \mathbf{B}_i^{(z)} + \mathbf{M}_i. \quad (6)$$

The matrix  $\mathbf{M}_i$  constructs a stall pattern of given size and maximum weight. It completely resolves minimal stall patterns, but it is not always successful for non-minimal stall patterns.

#### 3.2 Analysis of Bit-Flip Without Undetected Error Events

First assume that no undetected error events occur. By *undetected error event*, we refer to an *incorrectly* decoded component word with all-zero syndrome.

In general, for a non-minimal stall pattern of  $\mathbf{B}_i^{(z)}$  with masking matrices  $\mathbf{M}_i$  and  $\mathbf{M}_{i+1}$  as in Definition 1, all positions involved in the stall pattern are covered, but  $w_H(\mathbf{M}_i) + w_H(\mathbf{M}_{i+1}) = K \cdot L$ . This mask therefore reconstructs a stall pattern of correct size, but of maximum weight (compare (1)) which might introduce  $\bar{\epsilon}$  new errors, where

$$\bar{\epsilon} = K \cdot L - \epsilon. \quad (7)$$

For example, for the  $(4, 4)$  non-minimal stall pattern of Fig. 4, the bit-flip operation resolves the 12 erroneous bits of the stall pattern, but introduces 4 new errors, as indicated by red markers. These 4 new errors can then be corrected by a usual sliding-window decoding iteration.

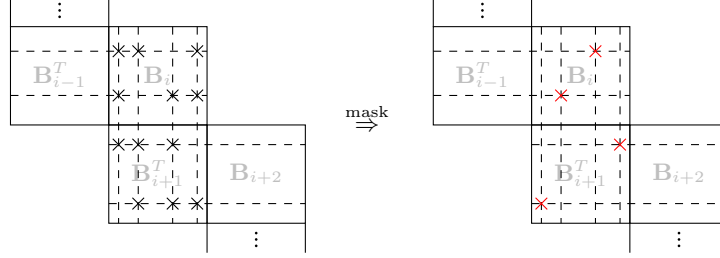


Fig. 4. Resolving of non-minimal stall pattern ( $t = 2$ )

**Theorem 1 (Guaranteed Resolving of Stall Patterns).** *Consider a staircase code with an extended BCH component code of minimum distance  $2t + 2$ . Assume that the sliding-window decoder has corrected all errors except for stall patterns with  $K, L < 2t + 2$ .*

*Then, the **bit-flip** operation from Definition 1 and a normal sliding-window iteration correct all these stall patterns if no undetected error events occur.*

*Proof.* When  $K, L < 2(t + 1)$ , the weight of every involved row is  $t + 1 \leq w_H(\mathbf{r}) \leq L$ , where the lower bound is given by the definition of a stall pattern. When the  $L$  involved bits of that row are flipped, the new weight of the row is

$$w_H(\bar{\mathbf{r}}) = L - w_H(\mathbf{r}) \leq (2(t + 1) - 1) - (t + 1) \leq t, \quad (8)$$

which can be corrected by the component codes in a normal sliding-window iteration.  $\square$

For larger  $K, L$ , the restriction of (8) no longer holds in general. Clearly, if  $\epsilon \geq d_{\min}^2$  it is possible that a stall pattern is undetectable. However, this case is unlikely and the staircase code cannot be protected against it. The more likely problem arises when  $K \geq 2(t + 1)$  and  $L \geq 2(t + 1)$ , but  $\epsilon < d_{\min}^2$ . Then the  $\epsilon$  previously error free positions can form a stall pattern of size  $(K, L)$  causing the bit-flip operation to result in another stall pattern. This case can be avoided with high probability by flipping only a single column if  $K \geq 2(t + 1)$  and  $L \geq 2(t + 1)$ . Then, with a high probability, some rows can be decoded, leading to decodable columns and eventually to resolving of the stall pattern.

### 3.3 Analysis of Bit-Flip With Undetected Error Events

Assume that undetected error events occur, i.e., there is an incorrect component word with all-zero syndrome. This component word is clearly a codeword of the component code, but the blocks do not give the correct staircase codeword. As a result, not all positions involved in a stall pattern can be located and (8) does not necessarily hold. When an extended BCH code of minimum distance  $2t + 2$  is used as component code, the lowest weight of a component error that can cause an undetected error event is  $t + 2$ . Since it is more likely that an

incorrect codeword is at distance  $t$  of the received word than at a smaller distance, performing iterations correcting less than  $t$  errors resolves many errors inserted due to undetected error events by the adjacent component code decoders, without being inserted again. However, multiple equal error vectors can prevent this, as the inserted errors appear in the same positions. It is therefore hard to give a theoretical analysis of stall patterns with undetected error events, but our simulations (Section 4.2) show that many of these patterns can still be resolved.

Fig. 5 shows an example of an uncorrectable  $(4, 3)$  stall pattern with three equal error vectors in the columns and additional undetected erroneous rows that cannot be resolved at all. While the decoder detects that the block is not valid, the columns cannot be located because their syndromes are zero and the operation *bit-flip* fails. In this case the stall pattern is neither detectable nor correctable.

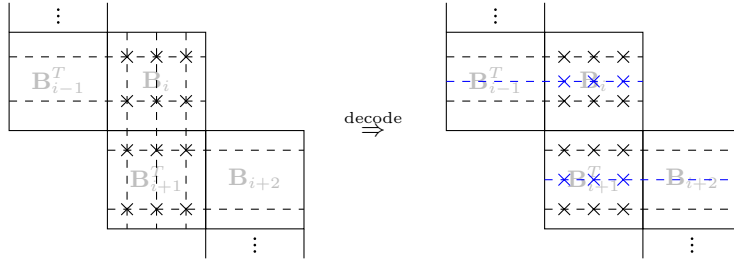


Fig. 5. Undetected error events in a  $(4, 3)$  stall patterns ( $d_{\min} = 6$ )

## 4 Improved Error-Floor Analysis

### 4.1 An Improved Analysis

The error floor of the standard sliding-window decoder (see Section 2.2) is dominated by minimal stall patterns [1]. As shown in Section 3.1, the improved decoder is able to solve all minimal stall patterns and its error floor is dominated by the remaining unsolvable stall patterns.

Since the success of the resolving strategy described in Section 3.1 depends on  $K, L$  and also  $\epsilon$ , we analyze each summand from (4) separately:

$$P_{C,old}(K, L, \epsilon) \triangleq \frac{\epsilon}{m^2} \cdot A_{K,L} \cdot \hat{N}_{K,L}^\epsilon \cdot (p + \xi)^\epsilon. \quad (9)$$

Since minimal stall patterns are no longer dominating, this estimation is not accurate, due to the overestimation of  $N_{K,L}^\epsilon \leq \hat{N}_{K,L}^\epsilon$ .

The problem of finding the number of stall patterns of weight  $\epsilon$  within  $K$  rows and  $L$  columns, is equivalent to the problem of finding the number of binary  $K \times L$  matrices with weight  $\epsilon$  and a given weight in each row and column. A solution to this combinatorial problem is given in [5]. The function takes a vector  $\mathbf{r} = [r_1, r_2, \dots, r_K]$  containing the row weights as well as  $\mathbf{s} = [s_1, s_2, \dots, s_L]$  containing the column weights and returns the number of distinct binary matrices that meet the weight restrictions on the rows and columns, denoted by  $\mathcal{A}(\mathbf{r}, \mathbf{s})$ . By definition, for stall patterns it holds that  $r_i \geq t + 1 \forall i \in$

$[1, K]$  and  $s_j \geq t + 1 \forall j \in [1, L]$ . Since every error has to be in one of the rows and one of the columns,  $\sum_{i=1}^K r_i = \epsilon$  and  $\sum_{j=1}^L s_j = \epsilon$ . Let  $\mathcal{R}$  be the set of all pairs of vectors, such that these conditions on  $\mathbf{r}$  and  $\mathbf{s}$  hold. The exact number of stall patterns of weight  $\epsilon$ , given the rows and columns, is

$$N_{K,L}^\epsilon = \sum_{r,s \in \mathcal{R}} \mathcal{A}(r, s) = D(\epsilon, K, L, 0), \quad (10)$$

where  $D(\cdot)$  is a function devised to recursively iterate through all vector pairs of  $\mathcal{R}$  and return the result of the summation (see Lemma 1 in the appendix). Thus, the contribution of stall patterns of a given size to the error floor can be stated without overestimating the number of unique stall patterns.

**Theorem 2.** *Consider a staircase code of block size  $m \times m$ . Given  $\xi$ , the contribution to the error floor stall patterns of size  $(K, L)$  and weight  $\epsilon$  is given by*

$$P_{C,new}(K, L, \epsilon) = \frac{\epsilon}{m^2} \cdot A_{K,L} \cdot N_{K,L}^\epsilon \cdot (p + \xi)^\epsilon. \quad (11)$$

The accuracy of (11) is determined by  $\xi$ , which is usually approximated by an estimation or simulation. This function determines the contribution to the error floor for all stall patterns, assuming they cannot be resolved by the techniques introduced in Section 3.1. Notice that from these “unsolvable” patterns, actually a large percentage *is* resolvable but due to the wide variety of combinations that can occur, it is difficult to obtain analytical results.

## 4.2 Simulation Results

To show the improvement in performance of our new technique, simulation results for a specific staircase code are presented. The parameters are chosen such that the encoder and decoder can employ a simpler structure than in [1], by using less memory and lower complexity component codes. The size of the blocks is quartered compared to [1] by using a  $[n = 510, k = 491]$  extended  $t = 2$  error correcting BCH code, resulting in block size  $\frac{n}{2} \times \frac{n}{2} = 255 \times 255$ . The corresponding rate for these parameters is  $R = \frac{n-m}{k-m} = \frac{236}{255}$ , which is slightly lower than the rate  $R_{[1]} = \frac{239}{255}$  of [1].

Decoding with the regular decoder, as described in Section 2.2, results in an error floor at  $\sim 2 \cdot 10^{-10}$  for  $p = 5 \cdot 10^{-3}$ , which is well above the desired error floor of  $10^{-15}$ . This error floor was found by simulation, using a sliding window of size  $W = 7$ . The variable  $\xi$  adjusting for the difference between estimated error floor and simulated probability of a stall pattern occurring was determined to be  $\xi = 1.6 \cdot 10^{-3}$ .

To find the capability of the improved decoder to solve stall patterns, a dedicated channel was implemented. Stall patterns of a certain size and weight are inserted at a random position within two blocks. At the output, it is determined whether the stall pattern was resolved or not. It was assumed, that the decoder was able to resolve all surrounding errors. For this reason and to avoid unwanted effects, such as resolving of a stall pattern through an undetected error event, no errors other than the ones belonging to the stall pattern were inserted. The simulation results are given in Table 1.

In Columns 1-3 of Table 1, the size and weight of the respective stall patterns are given. The fourth column gives the percentage of stall patterns that the

**Table 1.** Simulation and estimation results of the error floor when decoding staircase codes. The rightmost column shows the simulation results for our improved decoding method.

$K$	$L$	$\epsilon$	% Solved	$P_{C,old}$	$P_{C,new}$	$P_{floor,new}$
3	3	9	100%	$1.2 \cdot 10^{-10}$	$1.2 \cdot 10^{-10}$	0
3	4	12	51%	$4.1 \cdot 10^{-15}$	$4.1 \cdot 10^{-15}$	$2.0 \cdot 10^{-15}$
4	3	12	56%	$9.0 \cdot 10^{-15}$	$9.0 \cdot 10^{-15}$	$3.9 \cdot 10^{-15}$
4	4	12	100%	$1.4 \cdot 10^{-10}$	$1.3 \cdot 10^{-11}$	0
4	4	13	100%	$4.1 \cdot 10^{-12}$	$3.9 \cdot 10^{-13}$	0
4	4	14	79%	$4.4 \cdot 10^{-14}$	$2.0 \cdot 10^{-15}$	$4.4 \cdot 10^{-16}$
5	5	15	100%	$1.0 \cdot 10^{-10}$	$2.2 \cdot 10^{-12}$	0
5	5	16	99.9%	$7.5 \cdot 10^{-12}$	$1.7 \cdot 10^{-13}$	$6.8 \cdot 10^{-17}$
5	5	17	97.4%	$2.3 \cdot 10^{-13}$	$3.6 \cdot 10^{-15}$	$9.3 \cdot 10^{-17}$
5	5	18	95.1%	$4.4 \cdot 10^{-15}$	$3.4 \cdot 10^{-17}$	$1.7 \cdot 10^{-18}$
6	6	18	99.9%	$8.4 \cdot 10^{-11}$	$3.9 \cdot 10^{-13}$	$2.5 \cdot 10^{-16}$
6	6	19	99.9%	$1.0 \cdot 10^{-11}$	$6.2 \cdot 10^{-14}$	$7.9 \cdot 10^{-17}$
6	6	20	98.9%	$6.2 \cdot 10^{-13}$	$2.8 \cdot 10^{-15}$	$3.1 \cdot 10^{-17}$
7	7	21	100%	$7.3 \cdot 10^{-11}$	$7.5 \cdot 10^{-14}$	0
7	7	22	99.9%	$1.4 \cdot 10^{-11}$	$2.2 \cdot 10^{-14}$	$1.4 \cdot 10^{-17}$
7	7	23	99%	$1.3 \cdot 10^{-12}$	$1.7 \cdot 10^{-15}$	$1.8 \cdot 10^{-17}$

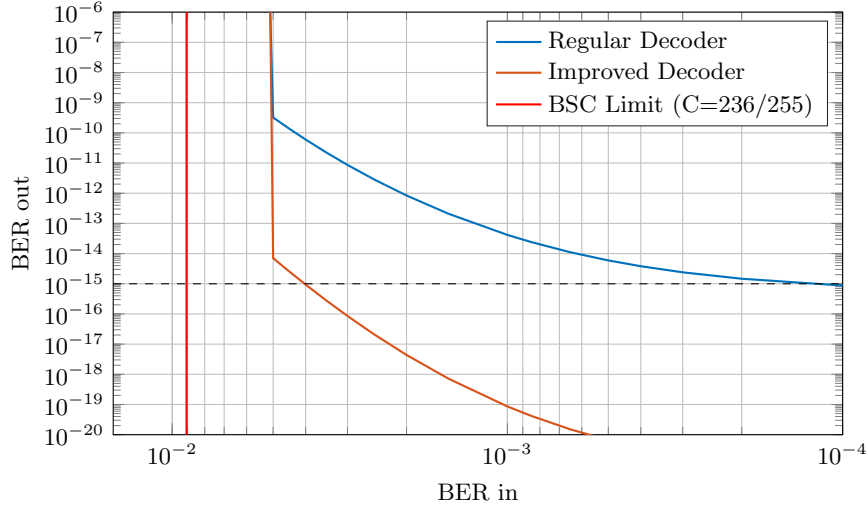
improved decoder was able to solve. For each size, more than 2000 stall patterns were inserted as described above. If any errors in the corresponding blocks were observed at the decoder output, the stall pattern was counted as unsolved. The remaining columns give the contribution to the error floor, as found by applying different estimations. The column labeled  $P_{C,old}$  gives the estimated contribution of stall patterns of the respective size, obtained by applying the estimation from [1], shown in (9). In column  $P_{C,new}$ , the corresponding values obtained by (11) are given. The column  $P_{floor,new}$  shows the contribution of the improved decoder which resolves the respective percentage of stall patterns. The new dominating contributors are the stall patterns of size (3, 4) and (4, 3), mainly due to the inability of the decoder to solve a large percentage of these. An illustration of such an unsolvable stall pattern was given in Fig. 5.

Our simulations show that the resolving strategy of only partially inverting large stall patterns (i.e., those which are not covered by Theorem 1) resolves a large percentage of these (e.g. 99.9% of (6, 6) stall patterns with  $\epsilon_{\min} = 18$ ).

The error floor of the staircase code with the given parameters is expected to be at  $\text{BER}_{out} \approx 9 \cdot 10^{-15}$  for a BSC crossover probability of  $p = 5 \cdot 10^{-3}$ . For comparison, the error floor of the same staircase code employing the regular decoder lies at  $\text{BER}_{out} \approx 2 \cdot 10^{-10}$ . The performance of the code in terms of the gap to BSC capacity for the given rate is estimated to 1dB and it achieves a net coding gain (NCG) of 9.16dB at a  $\text{BER}_{out}$  of  $10^{-15}$ . For comparison, the code presented in [1] achieves a NCG of 9.41dB and a gap to capacity of 0.56dB, however operating on much larger blocks.

Fig. 6 shows a comparison between the expected performance when using the regular decoder and the improved decoder. Note that the simulations on the ability of the improved decoder to resolve stall patterns were performed





**Fig. 6.** Performance comparison between improved and regular decoder

under the assumption that the decoder is able to isolate each stall pattern, i.e., resolve all errors which are not part of stall patterns. Furthermore, the assumption made in [1], that the stall patterns dominate the error floor, is adopted. Assuming that the window size of the regular decoder is chosen such that only the first (lowest indexed) block is free of all correctable errors, the sliding window size of the *improved* decoder has to be slightly larger to obtain a sufficient number of blocks that can be considered to be error-free with the exception of stall patterns.

## 5 Conclusion

Staircase codes are a powerful code construction for optical networks which perform close to the BSC capacity. The decoding based on a hard decision component code provides efficient implementations, even at high data rates. However, the usable block size is limited by requiring an error floor of  $10^{-15}$  in optical communications. In this work, an improved decoder is proposed, lowering the error floor significantly while increasing complexity only marginally. This improvement enables the use of smaller block sizes at comparable rates which effectively lowers the memory requirement and the component decoder complexity. Furthermore, an analysis of the error floor was presented resulting in a more accurate estimation, especially for the improved decoder.

Future work of interest includes an FPGA implementation capable of simulating the code with the given parameters, down to its estimated error floor to show that assumptions made on the capability of the decoder to correct errors surrounding a stall pattern hold.

## References

1. B. P. Smith, A. Farhood, A. Hunt, F. R. Kschischang and J. Lodge, *Staircase Codes: FEC for 100 Gb/s OTN*, CoRR, 2012

2. L. M. Zhang and F. R. Kschischang, *Staircase Codes With 6 % to 33 % Overhead*, Journal of Lightwave Technology, Vol. 32, No. 10, 2014
3. D. Gale, *A theorem on flows in networks*, Pacific Journal of Mathematics, Vol. 7, No. 2, 1957
4. C. Condo, F. Leduc-Primeau, G. Sarkis, P. Giard and W. J. Gross, *Stall Pattern Avoidance in Polynomial Product Codes*, CoRR, 2016
5. B. Wang and F. Zhang, *On the Precise Number of (0,1)-matrices in  $U(R,S)$* , Discrete Mathematics, 1998
6. S. Emmadi, K. R. Narayanan and H. D. Pfister, *Half-Product Codes for Flash Memory*, Non-Volatile Memories Workshop, 2015
7. T. Mittelholzer, T. Parnell, N. Papandreou and H. Pozidis, *Improving the error-floor performance of binary half-product codes*, International Symposium on Information Theory and Its Applications (ISITA), 2016
8. S. Sridharan, M. Jarchi and T. Coe, *Product code based forward error correction system*, US Patent App. 09/874,158, 2002
9. Y. Y. Jian, H. D. Pfister, K. R. Narayanan, Raghu Rao and R. Mazahreh, "Iterative hard-decision decoding of braided BCH codes for high-speed optical communication," 2013 IEEE Global Communications Conference (GLOBECOM), 2013

**Lemma 1.** Let  $\mathbf{A}_{K \times L}$  be a binary matrix of size  $K \times L$ . Denote by  $\mathbf{r}_1, \dots, \mathbf{r}_K$  the rows and by  $\mathbf{c}_1, \dots, \mathbf{c}_L$  the columns of the matrix. The cardinality of the set

$$\mathcal{Z}_{K,L}^\epsilon = \{\mathbf{A}_{K \times L} | w_H(\mathbf{A}) = \epsilon, w_H(\mathbf{r}_i) \geq t+1 \ \forall i \in [1, K], \\ w_H(\mathbf{c}_i) \geq t+1 \ \forall i \in [1, L]\}$$

is given by

$$|\mathcal{Z}_{K,L}^\epsilon| = D(\epsilon, K, L, 0) \quad (12)$$

with

$$D(a, b, c, d) = \sum_{w_{b+d} = \max\{t+1, a-c(b-1)\}}^{\min\{a-(b-1)(t+1), c\}} D(a - w_{b+d}, b-1, c, d)$$

and

$$D(w_{d+1}, 1, c, 0) = D(\epsilon, L, K, K), \quad (13)$$

$$D(w_{d+1}, 1, c, K) = \mathcal{A}([w_1, \dots, w_K], [w_{K+1}, \dots, w_{K+L}]). \quad (14)$$

*Proof.* In the first recursion step of (12) the number of errors  $w_K$  is assigned to row  $\mathbf{r}_K$ . The weight of the matrix is  $w_H(\mathbf{A}) = w_H(\mathbf{r}_1) + \dots + w_H(\mathbf{r}_K) = \epsilon = a$ . Obviously the weight of a row of a  $K \times L = b \times c$  matrix is upper bounded by  $w_H(\mathbf{r}) \leq c$ . It follows that  $w_H(\mathbf{r}_K) \leq \epsilon - L(K-1) = a - c(b-1)$ . By definition  $w_H(\mathbf{r}_K) = w_K \geq t+1$ . As this condition has to hold for each of the following  $K-1 = b-1$  rows, at most  $\epsilon - (K-1)(t+1) = a - (b-1)(t+1)$  can be in the first row. Combining these conditions gives the sum limits. The problem is reduced to distributing the remaining weight  $\epsilon - w_H(\mathbf{r}_K) = a - w_K$  among the rows of a  $K-1 \times L = b-1 \times L$  matrix, where the sum limits guarantee that the conditions on the rows can be met. When  $\mathbf{r}_1$  is reached (13), the recursion giving the weight  $w_{K+1}, \dots, w_{K+L}$  of the  $L$  columns is initiated, where  $d = K$  gives the offset in the weight vector. The same arguments as above hold for the limits of the sum. When the last column is reached (14), i.e.  $b = 1$  in the second recursion, the function  $\mathcal{A}([w_1, \dots, w_K], [w_{K+1}, \dots, w_{K+L}])$  from [5] gives the amount of matrices corresponding to the unique weight vector  $\mathbf{w}$  of size  $K+L$ .  $\square$